

Package: SemNetCleaner (via r-universe)

October 14, 2026

Title An Automated Cleaning Tool for Semantic and Linguistic Data

Version 2.0.0

Date 2026-09-12

Maintainer Alexander P. Christensen <alexpaulchristensen@gmail.com>

Description Implements several functions that automates the cleaning and spell-checking of text data. Also converges, finalizes, removes plurals and continuous strings, and puts text data in binary format for semantic network analysis. Uses the 'SemNetDictionaries' package to make the cleaning process more accurate, efficient, and reproducible.

License GPL (>= 3.0)

URL <https://github.com/AlexChristensen/SemNetCleaner>

BugReports <https://github.com/AlexChristensen/SemNetCleaner/issues>

NeedsCompilation no

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0), SemNetDictionaries (>= 0.1.8)

Imports foreign, future, future.apply, hunspell, phonics, progressr, R.matlab, readxl, searcher, stringi, stringdist

Suggests DT, htmlTable, knitr, markdown, promises, rmarkdown, rstudioapi, shiny (>= 1.8.0), tcltk, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev

Repository <https://alexchristensen.r-universe.dev>

Date/Publication 2026-09-13 16:46:34 UTC

RemoteUrl <https://github.com/alexchristensen/semnetcleaner>

RemoteRef HEAD

RemoteSha 60fe82de86fc642bf652f5d3032f562a6450a0af

Contents

SemNetCleaner-package	2
advance_wizard	3
automated_edit_server	4
automated_edit_ui	5
bad_response	5
best_guess	6
binary_to_responses	7
build_answer_burst	8
convert2snafu	9
finalize_textcleaner	11
letter.freq	12
make_shiny_search_fn	12
open.animals	13
open.clean	14
open.preprocess	14
pluralize	15
prepare_automated_for_review	16
qwerty.dist	17
read.data	18
read_uploaded_file	20
resp2bin	21
singularize	22
textcleaner	23
textcleaner_shiny	27
wizard_menu_labels	27
wizard_response_letters	28
Index	29

SemNetCleaner-package *SemNetCleaner-package*

Description

Implements several functions that automates the cleaning and spell-checking of text data. Also converges, finalizes, removes plurals and continuous strings, and puts text data in binary format for semantic network analysis. Uses the SemNetDictionaries package to make the cleaning process more accurate, efficient, and reproducible.

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

See Also

Useful links:

- <https://github.com/AlexChristensen/SemNetCleaner>
- Report bugs at <https://github.com/AlexChristensen/SemNetCleaner/issues>

 advance_wizard

Advance the textcleaner_shiny() Wizard by One Decision

Description

Wraps the mechanics common to every `textcleaner_shiny` wizard button click: builds a one-shot queued reader out of answers (typically from `build_answer_burst`), resumes `textcleaner(continue = ...)` with it, and returns whatever it stops at next – either a further `$stop = TRUE/$pending` decision, or the finished result.

Exported (not just internal) because `inst/Shiny/`'s own module files are sourced into the global environment by Shiny's own app-loading mechanism, not into the package namespace – an unexported function is not reachable from there by its bare name.

Usage

```
advance_wizard(
  continue,
  answers = character(),
  search_fn = searcher::search_site,
  edit_fn = function(x) x,
  cores = 1
)
```

Arguments

continue	List. The current <code>textcleaner()/spellcheck_dictionary()</code> return value to resume from (its own <code>\$stop = TRUE/\$pending</code> state)
answers	Character vector. The answers to queue for this one resume call, typically built by <code>build_answer_burst</code> . Defaults to <code>character()</code> (a resume with no real answer queued – immediately stops again at the same decision, e.g. after only a GOOGLE WORD/STRING/IT click)
search_fn	Function. Passed through to <code>textcleaner()</code> 's own <code>search_fn</code> . Defaults to <code>searcher::search_site</code> ; <code>textcleaner_shiny()</code> itself passes <code>make_shiny_search_fn</code> 's closure instead
edit_fn	Function. Passed through to <code>textcleaner()</code> 's own <code>edit_fn</code> . Defaults to a no-op (<code>function(x) x</code>), deferring the real automated-corrections review to <code>textcleaner_shiny()</code> 's own later step
cores	Numeric. Passed through to <code>textcleaner()</code> 's own <code>cores</code> . Defaults to 1

Value

Whatever `textcleaner()` returns for this resume call – a further `$stop = TRUE/$pending` decision, or the finished `textcleaner`-classed result

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

`automated_edit_server` *Automated Spell-Check Review App Server*

Description

The server half of the review app used to hand-correct automated spell-check changes mid-`textcleaner` pipeline (see `automated_edit()`). Exposed as a reusable, namespaceable building block (Phase 11) so a larger app (`textcleaner_shiny()`) can embed it as one module of a bigger wizard instead of only launching it standalone.

Usage

```
automated_edit_server(automated, on_finish = shiny::stopApp, id = NULL)
```

Arguments

<code>automated</code>	Matrix or data frame. The automated spell-check changes to review – a "from" column plus one or more "to_#" columns. Closed over directly by the returned server function; no file round-trip involved
<code>on_finish</code>	Function. Called with the edited matrix once either "Done" button is pressed. Defaults to <code>shiny::stopApp</code> for real standalone use; the injection point <code>shiny::testServer()</code> tests (and a parent module, via its own <code>on_finish</code>) use to capture the produced matrix directly instead of depending on <code>stopApp()</code> 's own behavior
<code>id</code>	Character or NULL. A shiny module namespace ID, matching <code>automated_edit_ui</code> 's own <code>id</code> . Defaults to NULL (no namespacing, this function's original behavior) – when supplied, the server body is wrapped in <code>shiny::moduleServer(id, ...)</code>

Value

A `function(input, output, session)` suitable for `shiny::shinyApp()`'s server argument (when `id` is NULL), or a module server usable inside a larger app (when `id` is supplied)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

automated_edit_ui	<i>Automated Spell-Check Review App UI</i>
-------------------	--

Description

The UI half of the review app used to hand-correct automated spell-check changes mid-[textcleaner](#) pipeline (see `automated_edit()`). Exposed as a reusable, namespaceable building block (Phase 11) so a larger app (`textcleaner_shiny()`) can embed it as one step of a bigger wizard instead of only launching it standalone.

Usage

```
automated_edit_ui(id = NULL)
```

Arguments

id	Character or NULL. A shiny module namespace ID. Defaults to NULL, matching this function's original, non-namespaced behavior exactly (<code>shiny::NS(NULL)</code> is the identity function) – pass a real ID only when embedding this UI as one module inside a bigger app, alongside the matching id on automated_edit_server
----	--

Value

A shiny UI definition

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

bad_response	<i>Changes Bad Responses to NA</i>
--------------	------------------------------------

Description

Determines whether responses are good or bad. Bad responses are replaced with missing (NA). Good responses are returned

Usage

```
bad_response(word, ...)
```

```
bad.response(word, ...)
```

Arguments

word	Character. A word to be tested for whether it is bad
...	Vector. Additional responses to be considered bad

Value

If response is bad, then returns NA. If response is valid, then returns the response

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Bad response
bad_response(word = " ")

# Good response
bad_response(word = "hello")

# Make a good response bad
bad_response(word = "hello", "hello")

# Add additional bad responses
bad_response(word = "hello", c("hello", "world"))
```

best_guess

Makes Best Guess for Spelling Correction

Description

A wrapper function for the best guess of a spelling mistake based on the letters, the ordering of those letters, and the potential for letters to be interchanged. The **Damerau-Levenshtein distance** is used to guide inferences into what word the participant was trying to spell from a dictionary (see SemNetDictionaries)

Usage

```
best_guess(word, full_dictionary, dictionary = NULL, tolerance = 1)

best.guess(word, full.dictionary, dictionary = NULL, tolerance = 1)
```

Arguments

word	Character. A word to get best guess spelling options from dictionary
full_dictionary	Character vector. The dictionary to search for best guesses in. See SemNetDictionaries
dictionary	Character. A dictionary from SemNetDictionaries for monikers (enhances guessing)

tolerance Numeric. The distance tolerance set for automatic spell-correction purposes. This function uses the function `stringdist` to compute the Damerau-Levenshtein distance, which is used to determine potential best guesses

Unique words (i.e., $n = 1$) that are within the (distance) tolerance are automatically output as best guess responses. This default is based on Damerau's (1964) proclamation that more than 80% of all human misspellings can be expressed by a single error (e.g., insertion, deletion, substitution, and transposition). If there is more than one word that is within or below the distance tolerance, then these will be provided as potential options.

The recommended and default distance tolerance is `tolerance = 1`, which only spell corrects a word if there is only one word with a DL distance of 1.

full.dictionary Legacy name for 'full_dictionary'.

Value

The best guess(es) of the word

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7, 171-176.

Philips, L. (1990). Hanging on the metaphone. *Computer Language*, 7(12), 39-43.

Examples

```
# Misspelled "bombay"
best_guess("bomba", full_dictionary = SemNetDictionaries::animals.dictionary)
```

binary_to_responses *Binary Responses to Character Responses*

Description

Converts a binary response matrix into characters for each participant

Usage

```
binary_to_responses(binary_matrix, to_data_frame = FALSE)

bin2resp(rmat, to.data.frame = FALSE)
```

Arguments

<code>binary_matrix</code>	Matrix. A binarized response matrix of verbal fluency or linguistic data
<code>to_data_frame</code>	Boolean. Should output be a data frame where participants are columns? Defaults to FALSE. Set to TRUE to convert output to data frame
<code>rmat</code>	Legacy name for ‘binary_matrix’.
<code>to.data.frame</code>	Legacy name for ‘to_data_frame’.

Value

A list containing objects for each participant and their responses

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10), -c(1:3)]

if(interactive())
{
  # Clean and preprocess data
  clean <- textcleaner(open.animals[, -c(1:2)], partBY = "row", dictionary = "animals")

  # Change binary response matrix to word response matrix
  char_matrix <- binary_to_responses(clean$responses$binary)
}
```

build_answer_burst	<i>Build a Wizard Answer Burst</i>
--------------------	------------------------------------

Description

Builds the exact queued-answer burst one [textcleaner_shiny](#) wizard button click needs to feed a single resume call (`textcleaner(continue = ... , reader = <queue reader>, ...)`) all the way through the one decision it represents, without starving or overrunning the console wizard’s own prompts. See [advance_wizard](#).

Exported (not just internal) because `inst/Shiny/`’s own module files are sourced into the global environment by Shiny’s own app-loading mechanism, not into the package namespace – an unexported function is not reachable from there by its bare name.

Usage

```

build_answer_burst(
  menu_number,
  typed_text = NULL,
  add_to_dictionary = FALSE,
  full_dictionary = NULL,
  original_check = NULL
)

```

Arguments

menu_number	Character or numeric, length 1. The real, currently-displayed menu position (or letter, for GO BACK/HELP/ a response-option suggestion) to send – see wizard_menu_labels/ wizard_response_letters
typed_text	Character or NULL. The free text typed for a TYPE MY OWN WORD/STRING decision. NULL (the default) for every other decision
add_to_dictionary	Logical. Whether to answer "yes" to any dictionary-add prompt this decision triggers. Defaults to FALSE
full_dictionary	Character vector. The current full dictionary, used to determine how many (if any) dictionary-add prompts typed_text/original_check will trigger. Required whenever either of those is supplied
original_check	Character or NULL. The pending decision's own original text, supplied only when menu_number corresponds to KEEP ORIGINAL – see this function's own header comment for the real hang this closes

Value

A character vector: the answers to queue, in order

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

convert2snafu	<i>Converts textcleaner object to a SNAFU GUI format</i>
---------------	--

Description

Converts [textcleaner](#) object to a SNAFU GUI format (only works for fluency data)

Usage

```
convert2snafu(..., category, path, reader = readline)
```

Arguments

<code>...</code>	Matrix or data frame. A clean response matrices
<code>category</code>	Character. Category of verbal fluency data
<code>path</code>	Character. Directory to save the SNAFU-formatted .csv file to
<code>reader</code>	Function. The input reader used to prompt for the output file name. Defaults to <code>readline</code> . Exposed so this can be driven programmatically (e.g. in automated tests) instead of real console input

Details

The format of the file has 7 columns:

- `id` — Defaults to the row names of the inputted data
- `listnum` — The list number for the fluency category. Defaults to 0. Future implementations will allow more lists
- `category` — The verbal fluency category that is input into the `category` argument
- `item` — The verbal fluency responses for every participant
- `RT` — Response time. Currently not implemented. Defaults to 0
- `RTstart` — Start of response time. Currently not implemented. Defaults to 0
- `group` — Names of groups. Defaults to the names of the objects input into the function (`...`)

Value

A .csv file formatted for SNAFU

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

For SNAFU, see: Zemla, J. C., Cao, K., Mueller, K. D., & Austerweil, J. L. (2020). SNAFU: The Semantic Network and Fluency Utility. *Behavior Research Methods*, 1-19. <https://doi.org/10.3758/s13428-019-01343-w>

Examples

```
# Convert data to SNAFU
if(interactive())
{convert2snafu(open.clean, category = "animals", path = tempdir())}
```

finalize_textcleaner *Finalize a textcleaner() Result*

Description

Reviews automated spell-check corrections and builds the final cleaned/binary/behavioral output – the exact logic `textcleaner` itself runs once, internally, via a pure code motion (Phase 11). Exposed so `textcleaner_shiny`'s corrections-review module can re-run this same, already-tested logic a second time once the user's real edits (from the embedded `automated_edit_server`) are known, instead of duplicating it. Byte-identical behavior to `textcleaner()`'s own original inline block, covered by the full `test-textcleaner.R` suite unchanged.

Exported (not just internal) because `inst/Shiny/`'s own module files are sourced into the global environment by Shiny's own app-loading mechanism, not into the package namespace – an unexported function is not reachable from there by its bare name.

Usage

```
finalize_textcleaner(
  res,
  data,
  type,
  ids,
  reader = readline,
  edit_fn = automated_edit,
  spell_check = res
)
```

Arguments

<code>res</code>	List. The in-progress <code>textcleaner()</code> result, with <code>\$spellcheck\$correspondence/\$automated/\$manual</code> already populated
<code>data</code>	Matrix or data frame. The already-spell-checked response data (<code>res\$responses\$original</code> for <code>type = "fluency"</code> , <code>res\$data\$original</code> for <code>type = "free"</code>)
<code>type</code>	Character. "fluency" or "free", matching <code>textcleaner()</code> 's own type
<code>ids</code>	Character vector. Participant IDs (<code>row.names(data)</code> for <code>type = "fluency"</code> , <code>unique(data[, "ID"])</code> for <code>type = "free"</code>)
<code>reader</code>	Function. Passed through to <code>correct_changes()</code> . Defaults to <code>readline</code>
<code>edit_fn</code>	Function. The automated-corrections review function. Defaults to <code>automated_edit()</code>
<code>spell_check</code>	List. The error-path fallback return value, matching <code>textcleaner()</code> 's own original (pre-extraction) behavior exactly. Defaults to <code>res</code>

Value

A finished, `textcleaner`-classed result

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

letter.freq

Letter Frequencies Based on 40,000 Words

Description

A vector corresponding the frequency of letters across 40,000 words. Retrieved from: <http://pi.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html>

Usage

```
data(letter.freq)
```

Format

letter.freq (26-element numeric vector)

Examples

```
data("letter.freq")
```

make_shiny_search_fn

Embedded-Search search_fn Factory for textcleaner_shiny()

Description

Returns a search_fn-shaped closure (matching textcleaner()'s own search_fn contract) for [textcleaner_shiny](#)'s wizard: instead of opening a real browser window, it writes a DuckDuckGo results URL into store for the app's own search modal to render in an embedded iframe. See this file's own header comment for why a real Google results page cannot be embedded this way at all.

Exported (not just internal) because inst/Shiny/'s own module files are sourced into the global environment by Shiny's own app-loading mechanism, not into the package namespace – an unexported function is not reachable from there by its bare name.

Usage

```
make_shiny_search_fn(store)
```

Arguments

store Function. A shiny::reactiveVal() (or any function accepting one argument) to write list(query =, url =) into whenever the returned closure is called

Value

A function(query, site, rlang) suitable for textcleaner()'s own search_fn argument

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

open.animals

Openness and Verbal Fluency

Description

Raw Animals verbal fluency data ($n = 516$) from Christensen et al. (2018).

Usage

```
data(open.animals)
```

Format

open.animals (matrix 516 x 38)

Details

First column is a grouping variable ("Group") with 1 corresponding to low openness to experience and 2 to high openness to experience

Second column is the latent variable of openness to experience with Intellect items removed (see Christensen et al., 2018 for more details).

Third column is the ID variable for each participant.

Columns 4-38 are raw fluency data.

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Examples

```
data("open.animals")
```

open.clean	<i>Cleaned Response Matrices (Openness and Verbal Fluency)</i>
------------	--

Description

Cleaned response matrices for the Animals verbal fluency data ($n = 516$) from Christensen et al. (2018).

Usage

```
data(open.clean)
```

Format

open.clean (matrix, 516 x 35)

References

Christensen, A. P., Kenett, Y. N., Cotter, K. N., Beaty, R. E., & Silvia, P. J. (2018). Remotely close associations: Openness to experience and semantic memory structure. *European Journal of Personality*, 32, 480-492.

Examples

```
data("open.clean")
```

open.preprocess	<i>Preprocessed textcleaner Object (Openness and Verbal Fluency)</i>
-----------------	--

Description

Preprocessed [textcleaner](#) object for the Animals verbal fluency data ($n = 516$) from Christensen and Kenett (2020).

Usage

```
data(open.preprocess)
```

Format

open.preprocess (list, length = 4)

References

Christensen, A. P., & Kenett, Y. N. (2020). Semantic network analysis (SemNA): A tutorial on preprocessing, estimating, and analyzing semantic networks. *PsyArxiv*.

Examples

```
data("open.preprocess")
```

pluralize

Converts Words to their Plural Form

Description

A function to change words to their plural form. The rules for converting words to their plural forms are based on the grammar rules.

This function handles most special cases and some irregular cases (see examples) but caution is necessary. If no plural form is identified, then the original word is returned.

Usage

```
pluralize(word)
```

Arguments

word	A word
------	--------

Value

Returns the word in plural form, unless a plural form could not be found (then the original word is returned)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Handles any prototypical cases
"dogs"
pluralize("dog")

"foxes"
pluralize("fox")

"wolves"
pluralize("wolf")

"octopi"
pluralize("octopus")

"taxa"
pluralize("taxon")
```

```
# And most special cases:
"wives"
pluralize("wife")

"roofs"
pluralize("roof")

"photos"
pluralize("photo")

# And some irregular cases:
"children"
pluralize("child")

"teeth"
pluralize("tooth")

"mice"
pluralize("mouse")
```

```
prepare_automated_for_review
```

Prepare the Automated Corrections Matrix for Review

Description

Reconstructs the automated spell-check corrections matrix `correct_changes()` itself hands to its `edit_fn` – restoring matrix-ness and the correspondence-matrix row name for a single-row selection (see this file’s own header comment for the real bug this fixes), and padding with enough “to_#” columns for the widest multi-word “from” entry. Exposed as its own function (Phase 11) so [textcleaner_shiny](#)’s corrections-review module can render the exact same shape `correct_changes()` itself will later expect back from `edit_fn` – computing this independently (e.g. from the raw, un-padded `res$spellcheck$automated`) produces a narrower matrix than `correct_changes()`’s own internal copy, which errors with “non-conformable arrays” the moment they’re compared.

Exported (not just internal) because `inst/Shiny/`’s own module files are sourced into the global environment by Shiny’s own app-loading mechanism, not into the package namespace – an unexported function is not reachable from there by its bare name.

Usage

```
prepare_automated_for_review(res)
```

Arguments

<code>res</code>	List. The in-progress <code>textcleaner()</code> result, with <code>\$spellcheck\$automated/\$correspondence</code> already populated
------------------	---

Value

The padded automated-corrections matrix, ready to hand to `automated_edit()`/[automated_edit_server](#) (or any other `edit_fn`)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

`qwerty.dist`*QWERTY Distance for Same Length Words*

Description

Computes QWERTY distance for words that have the same number of characters. Distance is computed based on the number of keys a character is away from another character on a QWERTY keyboard

Usage

```
qwerty.dist(wordA, wordB)
```

```
qwerty_distance(word_a, word_b)
```

Arguments

<code>wordA</code>	Legacy name for ‘word_a’.
<code>wordB</code>	Legacy name for ‘word_b’.
<code>word_a</code>	Character. Word to be compared
<code>word_b</code>	Character. Word to be compared

Value

Numeric value for distance between `word_a` and `word_b`

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Identical values for Damerau-Levenshtein
stringdist::stringdist("big", "pig", method = "dl")

stringdist::stringdist("big", "bug", method = "dl")

# Different distances for QWERTY
qwerty_distance("big", "pig")
```

```
qwerty_distance("big", "bug") # Probably meant to type "bug"
```

read.data

Read in Common Data File Extensions

Description

A single function to read in common data file extensions. Note that this function is specialized for reading in text data in the format necessary for functions in SemNetCleaner

File extensions supported:

- .Rdata
- .rds
- .csv
- .xlsx
- .xls
- .sav
- .txt
- .mat
- .dat

Usage

```
read.data(file = file.choose(), header = TRUE, sep = ",", ...)
```

```
read_data(file = file.choose(), header = TRUE, sep = ",", ...)
```

Arguments

- | | |
|--------|--|
| file | Character. A path to the file to load. Defaults to interactive file selection using file.choose |
| header | Boolean. A logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: header is set to TRUE if and only if the first row contains one fewer field than the number of columns |
| sep | Character. The field separator character. Values on each line of the file are separated by this character. If sep = "" (the default for read.table) the separator is a 'white space', that is one or more spaces, tabs, newlines or carriage returns |
| ... | Additional arguments. Allows for additional arguments to be passed onto the respective read functions. See documentation in the list below: <ul style="list-style-type: none"> • .Rdata load • .rds readRDS |

- .csv `read.table`
- .xlsx `read_excel`
- .xls `read_excel`
- .sav `read.spss`
- .txt `read.table`
- .mat `readMat`
- .dat `read.table`

Value

A data frame containing a representation of the data in the file. If file extension is ".Rdata", then data will be read to the global environment

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

R Core Team

R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

readxl

Hadley Wickham and Jennifer Bryan (2019). readxl: Read Excel Files. R package version 1.3.1. <https://CRAN.R-project.org/package=readxl>

R.matlab

Henrik Bengtsson (2018). R.matlab: Read and Write MAT Files and Call MATLAB from Within R. R package version 3.6.2. <https://CRAN.R-project.org/package=R.matlab>

Examples

```
# Use this example for your data
if(interactive())
{read_data()}

# Example for CRAN tests
## Create test data
test1 <- c(1:5, "6,7", "8,9,10")

## Path to temporary file
tf <- tempfile()

## Create test file
writeLines(test1, tf)

## Read in data
read_data(tf)
```

```
# See documentation of respective R functions for specific examples
```

read_uploaded_file	<i>Read a File Uploaded Through Shiny</i>
--------------------	---

Description

Reads a file uploaded via `shiny::fileInput()` (Phase 11, `textcleaner_shiny()`'s own upload step). Shiny stores an uploaded file at a temporary path (`datapath`) that does not necessarily carry the original file's extension, but `read_data` dispatches purely by sniffing the path's own extension – this function bridges the two by renaming/copying `datapath` to one ending in the real, original extension (from `filename`, the name the browser reported) before handing off.

Usage

```
read_uploaded_file(datapath, filename, header = TRUE, sep = ",")
```

Arguments

<code>datapath</code>	Character. The temporary file path <code>shiny::fileInput()</code> stored the upload at (<code>input\$file\$datapath</code>)
<code>filename</code>	Character. The original file name the browser reported (<code>input\$file\$name</code>) – used only for its extension
<code>header</code>	Boolean. Passed through to <code>read_data</code> . Defaults to <code>TRUE</code>
<code>sep</code>	Character. Passed through to <code>read_data</code> . Defaults to <code>","</code>

Value

The data read from the uploaded file (a data frame for every supported extension except `.Rdata`, an unmodified round-trip)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

`resp2bin`*Converts a Response Matrix to a Binary Response Matrix*

Description

Converts a response matrix (verbal fluency or other linguistic data, one participant per row) into a binary (participant x vocabulary) response matrix

Usage

```
resp2bin(resp)
```

```
responses_to_binary(responses)
```

Arguments

`resp` Legacy name for ‘responses’.

`responses` Matrix or data frame. A response matrix of verbal fluency or linguistic data

Value

A binary matrix with one row per participant and one column per unique response observed across the whole sample

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Toy example
raw <- open.animals[c(1:10), -c(1:3)]

if(interactive())
{
  # Clean and preprocess data
  clean <- textcleaner(open.animals[, -c(1:2)], partBY = "row", dictionary = "animals")

  # Change response matrix to binary response matrix
  binary_matrix <- responses_to_binary(clean$responses$corrected)
}
```

`singularize`*Converts Words to their Singular Form*

Description

A function to change words to their singular form. The rules for converting words to their singular forms are based on the *inverse* of the grammar rules. This function handles most special cases and some irregular cases (see examples) but caution is necessary. If no singular form is identified, then the original word is returned.

Usage

```
singularize(word, dictionary = TRUE)
```

Arguments

<code>word</code>	Character. A word
<code>dictionary</code>	Boolean. Should dictionary be used to verify word exists? Default to TRUE

Value

Returns the word in singular form, unless a singular form could not be found (then the original word is returned)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Examples

```
# Handles any prototypical cases
# "dog"
singularize("dogs")

# "fox"
singularize("foxes")

# "wolf"
singularize("wolves")

# "octopus"
singularize("octopi")

# "taxon"
singularize("taxa")

# And most special cases:
# "wife"
singularize("wives")
```

```
# "fez"
singularize("fezzes")

# "roof"
singularize("roofs")

# "photo"
singularize("photos")

# And some irregular cases:
# "child"
singularize("children")

# "tooth"
singularize("teeth")

# "mouse"
singularize("mice")
```

textcleaner

Text Cleaner

Description

An automated cleaning function for spell-checking, de-pluralizing, removing duplicates, and binarizing text data

Usage

```
textcleaner(
  data = NULL,
  type = c("fluency", "free"),
  miss = 99,
  partBY = c("row", "col"),
  dictionary = NULL,
  spelling = c("UK", "US"),
  add.path = NULL,
  keepStrings = FALSE,
  allowPunctuations,
  allowNumbers = FALSE,
  lowercase = TRUE,
  keepLength = NULL,
  keepCue = FALSE,
  continue = NULL,
  reader = readline,
  search_fn = searcher::search_site,
  cores = 1,
```

```
    edit_fn = automated_edit
  )
```

Arguments

data Matrix or data frame.
For type = "fluency", data are expected to follow wide formatting (IDs are the row names and are **not** a column in the matrix or data frame):

row.names	Response 1	Response 2	Response n
ID_1	1	2	n
ID_2	1	2	n
ID_n	1	2	n

For type = "free", data are expected to follow long formatting:

ID	Cue	Response
1	1	1
1	1	2
1	1	n
1	2	1
1	2	2
1	2	n
1	n	1
1	n	2
1	n	n
2	1	1
2	1	2
2	1	n
2	2	1
2	2	2
2	2	n
2	n	1
2	n	2
2	n	n
n	1	1
n	1	2
n	1	n
n	2	1
n	2	2
n	2	n
n	n	1
n	n	2
n	n	n

type Character vector. Type of task to be preprocessed.

- "fluency" — Verbal fluency data (e.g., categories, phonological, synonyms)
- "free" — Free association data (e.g., cue terms or words)

miss	Numeric or character. Value for missing data. Defaults to 99
partBY	Character. Are participants by row or column? Set to "row" for by row. Set to "col" for by column
dictionary	Character vector. Can be a vector of a corpus or any text for comparison. Dictionary to be used for more efficient text cleaning. Defaults to NULL, which will use general.dictionary Use <code>dictionaries()</code> or <code>find.dictionaries()</code> for more options (See <code>SemNetDictionaries</code> for more details)
spelling	Character vector. English spelling to be used. • "UK" — For British spelling (e.g., colour, grey, programme, theatre) • "US" — For American spelling (e.g., color, gray, program, theater)
add.path	Character. Path to additional dictionaries to be found. DOES NOT search recursively (through all folders in path) to avoid time intensive search. Set to "choose" to open an interactive directory explorer
keepStrings	Boolean. Should strings be retained or separated? Defaults to FALSE. Set to TRUE to retain strings as strings
allowPunctuations	Character vector. Allows punctuation characters to be included in responses. Defaults to "-". Set to "all" to keep all punctuation characters
allowNumbers	Boolean. Defaults to FALSE. Set to TRUE to keep numbers in text
lowercase	Boolean. Should words be converted to lowercase? Defaults to TRUE. Set to FALSE to keep words as they are
keepLength	Numeric. Maximum number of words allowed in a response. Defaults to NULL. Set a number to keep responses with words less than or equal to the number (e.g., 3 will keep responses with three or less words)
keepCue	Boolean. Should cue words be retained in the responses? Defaults to FALSE. Set to TRUE to allow cue words to be retained
continue	List. A result previously unfinished that still needs to be completed. Allows you to continue to manually spell-check their data after you've closed or errored out. Also accepts a previously completed result (e.g., a saved return value of this same function, or of calling the manual spell-check stage directly) to skip straight to finalizing the output without repeating the spell-check. Defaults to NULL
reader	Function. The input reader used for every interactive prompt in the manual spell-check wizard. Defaults to <code>readline</code> . Exposed so the whole wizard can be driven programmatically (e.g. in automated tests) with a pre-scripted queue of answers instead of real console input
search_fn	Function. The function used to open a browser search for the "GOOGLE WORD"/"GOOGLE STRING" manual spell-check options. Defaults to <code>searcher::search_site</code>
cores	Numeric. Number of computer cores to use in parallelized computations (the "ad hoc" moniker re-check after manual spell-check). Defaults to 1 (sequential)
edit_fn	Function. The spreadsheet-editor function used to review automated spell-check changes. Defaults to <code>automated_edit</code> , a shiny/DT-based review app (requires those packages to be installed). Pass <code>edit_fn = edit</code> to use base R's lightweight <code>edit()</code> instead, with no extra dependencies. Exposed so this step can be driven programmatically instead of opening a real interactive editor

Value

This function returns a list containing the following objects:

binary	A matrix of responses where each row represents a participant and each column represents a unique response. A response that a participant has provided is a '1' and a response that a participant has not provided is a '0'
responses	A list containing two objects: • clean — A response matrix that has been spell-checked and de-pluralized with duplicates removed. This can be used as a final dataset for analyses (e.g., fluency of responses) • original — The original response matrix that has had white spaces before and after words response. Also converts all upper-case letters to lower case
spellcheck	A list containing three objects: • full — All responses regardless of spell-checking changes • auto — Only the incorrect responses that were changed during spell-check
removed	A list containing two objects: • rows — Identifies removed participants by their row (or column) location in the original data file • ids — Identifies removed participants by their ID (see argument data)
partChanges	A list where each participant is a list index with each response that was been changed. Participants are identified by their ID (see argument data). This can be used to replicate the cleaning process and to keep track of changes more generally. Participants with NA did not have any changes from their original data and participants with missing data are removed (see removed\$ids)

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

References

Christensen, A. P., & Kenett, Y. N. (in press). Semantic network analysis (SemNA): A tutorial on preprocessing, estimating, and analyzing semantic networks. *Psychological Methods*.

Hornik, K., & Murdoch, D. (2010). Watch Your Spelling!. *The R Journal*, 3, 22-28.

Examples

```
# Toy example
raw <- open.animals[c(1:10),-c(1:3)]

if(interactive())
{
  #Full test
  clean <- textcleaner(open.animals[, -c(1,2)], partBY = "row", dictionary = "animals")
}
```

textcleaner_shiny	<i>Point-and-Click textcleaner() Wizard</i>
-------------------	---

Description

Launches a Shiny app wrapping the entire [textcleaner](#) pipeline end to end – upload, options, automated spell-check, manual spell-check wizard, automated-corrections review, and download – as a point-and-click alternative to the interactive console wizard. Every menu option becomes a real clickable button (plus a free-text box for typing a custom word/string directly); the "GOOGLE WORD"/"GOOGLE STRING"/"GOOGLE IT" options open an embedded in-app search instead of a new browser window.

Usage

```
textcleaner_shiny(...)
```

Arguments

... Additional arguments passed to `shiny::runApp()` (e.g. `port`, `launch.browser`).

Value

Called for its side effect (launching the Shiny app). Nothing is returned to the calling R session – results are downloaded from within the app itself (see the app's own Export tab).

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

wizard_menu_labels	<i>Wizard Menu Labels</i>
--------------------	---------------------------

Description

The exact, position-ordered menu labels `spellcheck_menu()` itself displays for a given context/keep_strings combination – used by [textcleaner_shiny](#)'s wizard module to render one button per meta-option, each labeled with its real text and sending its real 1-indexed display position as the raw answer (see [build_answer_burst](#)).

Exported (not just internal) because `inst/Shiny/`'s own module files are sourced into the global environment by Shiny's own app-loading mechanism, not into the package namespace – an unexported function is not reachable from there by its bare name.

Usage

```
wizard_menu_labels(context, keep_strings)
```

Arguments

context	Character vector or NULL. The pending decision's own <code>\$pending\$context</code> – NULL for a single-response decision, the multi-word vector otherwise
keep_strings	Logical. The pending decision's own <code>\$pending\$keep_strings</code>

Value

A character vector of menu labels, in real display order

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

wizard_response_letters

Wizard Response Letters

Description

The QWERTY letters `appropriate_answer()` assigns to non-default choices (possible, i.e. `best_guess`'s own candidate list) in display order – used by `textcleaner_shiny`'s wizard module to label its suggestion buttons and to build the right raw answer for one (see `build_answer_burst`).

Exported (not just internal) for the same reason as `wizard_menu_labels` – reachability from `inst/Shiny/`.

Usage

```
wizard_response_letters(n)
```

Arguments

n	Numeric. The number of candidates to label (<code>length(pending\$possible)</code>). Explicitly capped at 10 here (not just assumed) – <code>best_guess()</code> itself never returns more than 10 candidates in its give-up list, but indexing a 10-element vector with <code>seq_len(n)</code> for $n > 10$ would otherwise silently return n elements, the extra ones NA, rather than truncating
---	---

Value

A character vector of QWERTY letters, in display order

Author(s)

Alexander Christensen <alexpaulchristensen@gmail.com>

Index

* datasets

- letter.freq, [12](#)
- open.animals, [13](#)
- open.clean, [14](#)
- open.preprocess, [14](#)

advance_wizard, [3](#), [8](#)
automated_edit_server, [4](#), [5](#), [11](#), [17](#)
automated_edit_ui, [4](#), [5](#)

bad.response (bad_response), [5](#)
bad_response, [5](#)
best.guess (best_guess), [6](#)
best_guess, [6](#), [28](#)
bin2resp (binary_to_responses), [7](#)
binary_to_responses, [7](#)
build_answer_burst, [3](#), [8](#), [27](#), [28](#)

convert2snafu, [9](#)

file.choose, [18](#)
finalize_textcleaner, [11](#)

general.dictionary, [25](#)

letter.freq, [12](#)
load, [18](#)

make_shiny_search_fn, [3](#), [12](#)

open.animals, [13](#)
open.clean, [14](#)
open.preprocess, [14](#)

pluralize, [15](#)
prepare_automated_for_review, [16](#)

qwerty.dist, [17](#)
qwerty_distance (qwerty.dist), [17](#)

read.data, [18](#)
read.spss, [19](#)

read.table, [18](#), [19](#)
read_data, [20](#)
read_data (read.data), [18](#)
read_excel, [19](#)
read_uploaded_file, [20](#)
readMat, [19](#)
readRDS, [18](#)
resp2bin, [21](#)
responses_to_binary (resp2bin), [21](#)

SemNetCleaner (SemNetCleaner-package), [2](#)
SemNetCleaner-package, [2](#)
singularize, [22](#)
stringdist, [7](#)

textcleaner, [4](#), [5](#), [9](#), [11](#), [14](#), [23](#), [27](#)
textcleaner_shiny, [3](#), [8](#), [11](#), [12](#), [16](#), [27](#), [27](#), [28](#)

wizard_menu_labels, [9](#), [27](#), [28](#)
wizard_response_letters, [9](#), [28](#)